Live Chat App — Architecture & Flow Document

Overview

A real-time chat application where users can log in with a unique username, send and receive live messages using WebSockets, and view past messages. Features include a search bar for message filtering and a toggleable light/dark theme.

---

1. Architecture Summary

Frontend (React)

- Built with **React.js**
- Handles:
    - User login UI
    - Real-time chat interface
    - Message search
    - Theme toggle (light/dark)
- Communicates with backend via:
    - **Axios (HTTP)** – for login and fetching message history
    - **Socket.IO-client (WebSocket)** – for real-time messaging

Backend (Node.js + Express + Socket.IO)

- Built with **Node.js + Express**
- Stores data **in-memory**
- Responsibilities:
    - Validates unique usernames via POST /login
    - Serves message history via GET /messages
    - Handles WebSocket events using **Socket.IO**
    - Broadcasts new messages to all connected clients

Storage

- Users and messages are stored **temporarily in memory**:

```
const users = {};     // { "alice": true }
const messages = [];  // [{ username, text }]
```

2. Application Flow

User Login

1. User enters a unique name on the frontend.
2. React app sends POST /login to backend.
3. Backend checks if the username is taken:
    - If not, adds to in-memory users object.
    - Responds with { username } if valid.
4. Client stores the username and connects to Socket.IO with socket.emit("join").

Message History

1. On successful login, frontend sends GET /messages.
2. Backend responds with all chat history from messages[].
3. Frontend renders the messages in chat window.

Real-Time Messaging

1. User types and sends a message.
2. Frontend emits sendMessage event via Socket.IO:

```
socket.emit("sendMessage", { username, text });
```

    - Appends message to in-memory array.
    - Broadcasts to all clients using:

```
io.emit("receiveMessage", message);
```

4. All connected clients receive and display the new message.

Message Search (Frontend only)

- Controlled input filters messages[] by matching username or text.
- No extra backend queries; purely handled in React using .filter().

Theme Toggle (Frontend only)

- Users can switch between light and dark themes.

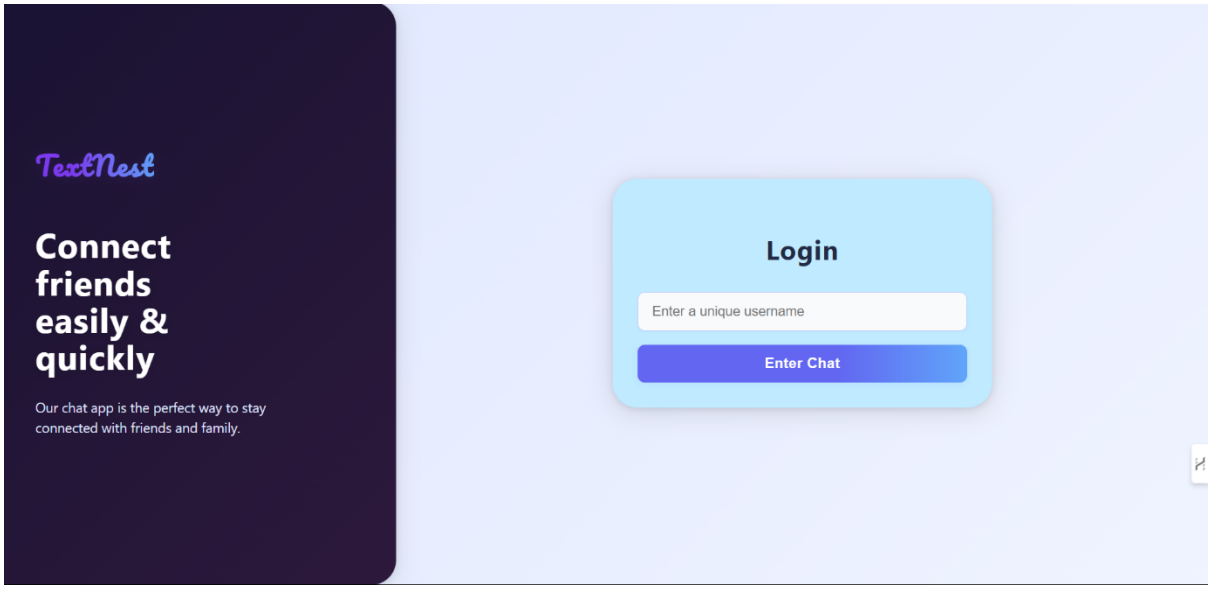
- Uses useState and conditional CSS classes (e.g., light-theme, dark-theme).
- Optionally stored in localStorage.

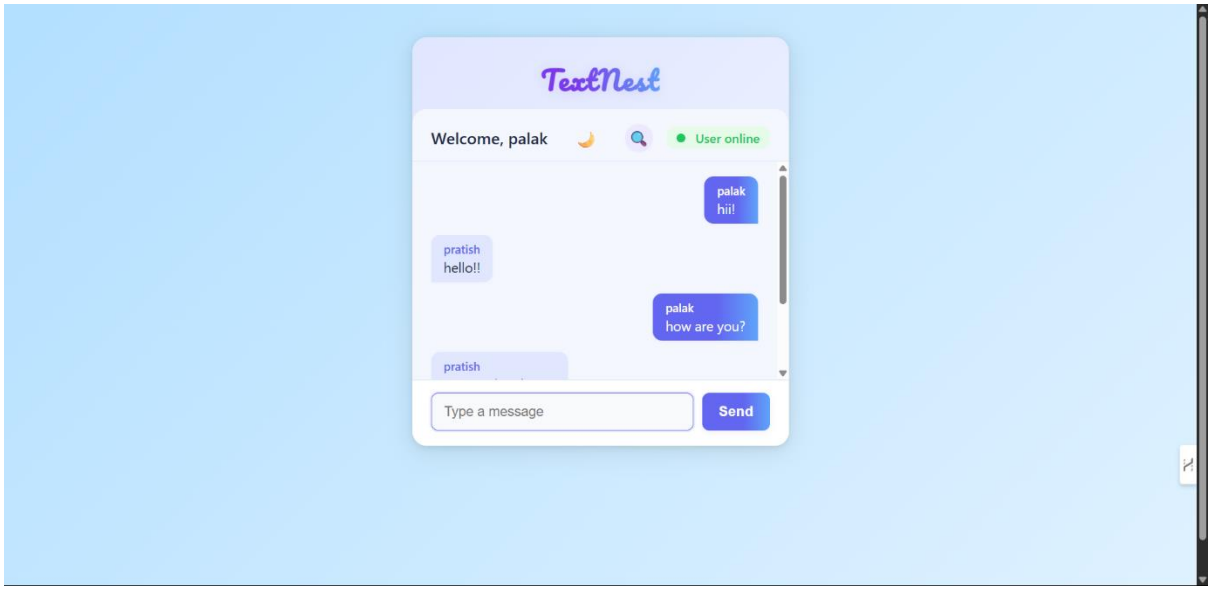
Disconnection

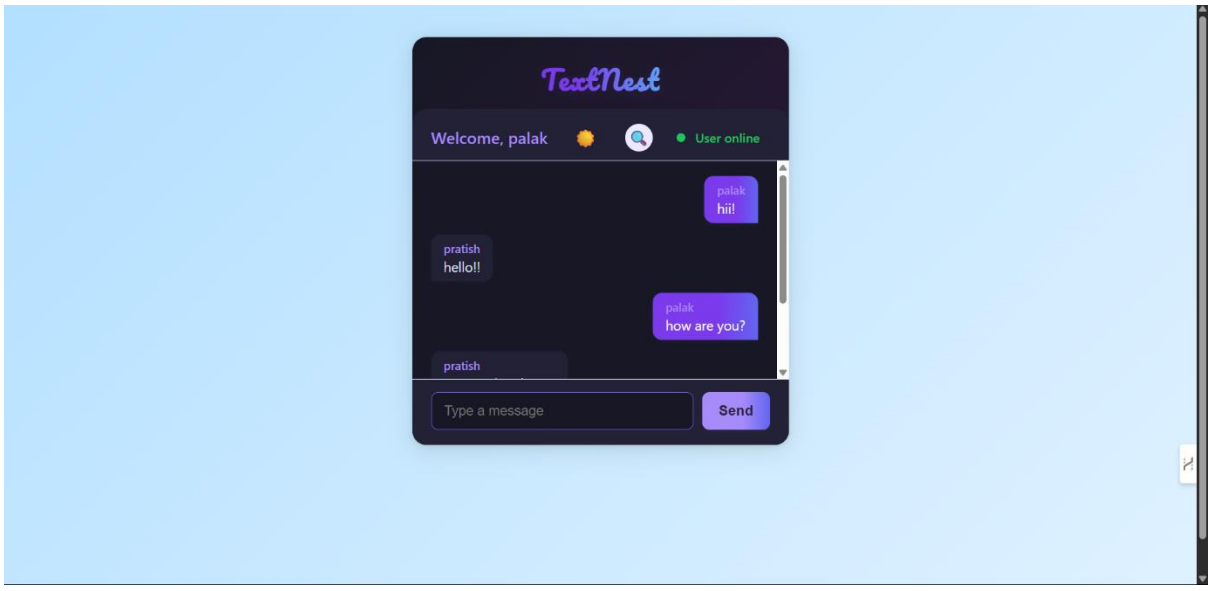
- When user leaves or refreshes, backend logs disconnection.
- No session persistence; user and messages are kept only in RAM and are lost on server restart.

Limitations

- No persistent storage (data lost on server restart)
- No authentication beyond username uniqueness
- No user sessions or logout
- No message timestamps (unless manually added)


TextNest
Connect friends easily & quickly
Our chat app is the perfect way to stay connected with friends and family.
Login
Enter a unique username
Enter Chat


TextNest
Welcome, palak
User online
palak
hii!
pratish
hello!!
palak
how are you?
pratish
Type a message
Send


TextNest
Welcome, palak
User online
palak
hii!
pratish
hello!!
palak
how are you?
pratish
Type a message
Send